Practical 1:-
Data Wrangling, I
Perform the following operations using Python on any open source dataset (e.g., data.csv)
1. Import all the required Python Libraries.
2. Locate an open source data from the web (e.g. https://www.kaggle.com). Provide a clear description of the data and its source (i.e., URL of the web site).
3. Load the Dataset into pandas data frame.
4. Data Preprocessing: check for missing values in the data using pandas isnull(), describe() function to get some initial statistics. Provide variable descriptions. Types of variables etc. Check the dimensions of the data frame.
5. Data Formatting and Data Normalization: Summarize the types of variables by checking the data types (i.e., character, numeric, integer, factor, and logical) of the variables in the data set. If variables are not in the correct data type, apply proper type conversions.
6. Turn categorical variables into quantitative variables in Python.
In addition to the codes and outputs, explain every operation that you do in the above steps and explain everything that you do to import/read/scrape the data set

```
import pandas as pd
import numpy as np

df1 = {
'Name':['George','Andrea','micheal','maggie','Ravi','Xien','Jalpa',np.nan],
'State':['Arizona','Georgiaprint(df1)print(df1)','Newyork','Indiana','Florida','California',np.nan,np.nan],
'Gender':["M","F","M","F","M","M",np.nan,np.nan],
'Score':[63,48,56,75,np.nan,77,np.nan,np.nan]
}

df1 = pd.DataFrame(df1,columns=['Name','State','Gender','Score'])

print(df1)

df1.shape

df1.info()

df1.describe()

df1.isnull()

df1.isnull().sum()

df1 = df1.dropna(subset=["Name"])

df1.info()

df1.head(2)

df1.select_dtypes(exclude='object').dtypes

df1.select_dtypes(include='object').dtypes

df1.tail()
```

Practical 2:-

Data Wrangling II

Create an “Academic performance” dataset of students and perform the following operations using Python.

1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them.

2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.

3. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution.

Reason and document your approach properly.

```
import numpy as np
import pandas as pd

df = pd.read_csv("xAPI-Edu-Data.csv")

df

df.head()

df.tail()

df.describe()

df.info()

df.shape

df.isnull().any().any()

df.isnull().sum()

avg_val = df["Discussion"].astype("float").mean()
avg_val

df["Discussion"].replace(np.NaN, avg_val, inplace=False)

df.isnull().sum()
```

Practical 3:-

Descriptive Statistics - Measures of Central Tendency and variability

Perform the following operations on any open source dataset (e.g., data.csv)

1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variable. For example, if your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable.

```
import numpy as np
import pandas as pd
import statistics as st

df = pd.read_csv("Mall_Customers.csv")
df

df.mean

df.loc[:,'Age'].mean

df.median

df.loc[:,'Age'].median()

df.mode()

df.loc[:,'Age'].mode()

df.min()

df.loc[:,'Age'].min(skipna = False)

df.max()

df.loc[:,'Age'].max(skipna = False)

df.std

df.loc[:,'Age'].std()

df.groupby(['Gender'])['Age'].mean()

df_u=df.rename(columns= {'Annual Income (k$)':'Income'}, inplace= False)
df_u

df_u.groupby(['Gender']).Income.mean()

from sklearn import preprocessing
enc = preprocessing.OneHotEncoder()
enc_df = pd.DataFrame(enc.fit_transform(df[['Gender']]).toarray())
enc_df
df_encode =df_u.join(enc_df)
df_encode
```

2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of ‘Iris-setosa’, ‘Iris-versicolor’ and ‘Iris- versicolor’ of iris.csv dataset.
Provide the codes with outputs and explain everything that you do in this step.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df_iris = pd.read_csv("Iris.csv")
df_iris.head()

print('Iris-setosa')
setosa = df_iris['Species'] == 'Iris-setosa'
print(df_iris[setosa].describe())
print('\nIris-versicolor')
versicolor = df_iris['Species'] == 'Iris-versicolor'
print(df_iris[versicolor].describe())
print('\nIris-virginica')
virginica = df_iris['Species'] == 'Iris-virginica'
print(df_iris[virginica].describe())

df_iris.dtypes.value_counts()
```

Practical 4:-
Data Analytics I
Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset.
The objective is to predict the value of prices of the house using the given features.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("boston.csv")
data.head()

X=data['dis'].values
Y=data['medv'].values
Y

mean_x=np.mean(X)
mean_y=np.mean(Y)
n=len(X)
n

numer=0
denom=0
for i in range(n):
   numer+=(X[i]-mean_x)*(y[i]-mean_y)
   denom+=(X[i]-mean_x)**2
b1=numer/denom
b0=mean_y-(b1*mean_x)
print("Coeeficients")
print("m=",b1)
print("c=",b0)

max_x=np.max(X)
min_x=np.min(X)
x=np.linspace(min_x,max_x,1000)
y=b0+b1*x
plt.plot(x,y,color='green',label='Regression Line')
plt.scatter(X,Y,c='red',label='Scatter Plot')
plt.xlabel('Head Size in cm3')
plt.ylabel('Brain Weight in grams')
plt.legend()
plt.show()
```

Practical 5:-
Data Analytics II
1. Implement logistic regression using Python/R to perform classification onSocial_Network_Ads.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(random_state = 0)
log_reg.fit(X_train, y_train)
y_pred = log_reg.predict(X_test)
y_pred

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm

from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
   plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],  color = ListedColormap(('red', 'green'))(i), label = j)
```

Practical 6:-
Data Analytics III
1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

```
import pandas as pd
from matplotlib.pyplot as plt

df=pd.read_csv("iris.csv")
df.head(10)

x=df.iloc[:,0:4]
y=df.iloc[:,-1]
y

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,random_state=1)
x_test

from sklearn.naive_bayes import GaussianNB
model=GaussianNB()
model.fit(x_train,y_train)
y_predicted=model.predict(x_test)
y_predicted

from sklearn.metrics import confusion_matrix,classification_report
cm=confusion_matrix(y_test,y_predicted)
cm

cl_report=classification_report(y_test,y_predicted)
cl_report

cm_df=pd.DataFrame(cm,
            index=['SETOSA','VERSICOLR','VIRGINICA'],
            columns=['SETOSA','VERSICOLR','VIRGINICA'])
cm_df

from matplotlib import pyplot as plt
plt.figure(figsize=(5,4))
sns.heatmap(cm_df,annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actual Values')
plt.xlabel('Predicated Values')
plt.show()

def accuracy_cm(tp,fn,fp,tn):
  return (tp+tn)/(tp+fp+tn+fn)
def precision_cm(tp,fn,fp,tn):
  return tp/(tp+fp)
def recall_cm(tp,fn,fp,tn):
  return tp/(tp+fn)
def f1_score(tp,fn,fp,tn):
  return (2/(1/recall_cm(tp,fn,fp,tn))+precision_cm(tp,fn,fp,tn))
def error_rate_cm(tp,fn,fp,tn):
  return 1-accuracy_cm(tp,fn,fp,tn)

tp = cm[2][2]
```

```
fn = cm[2][0]+cm[2][1]
fp = cm[0][2]+cm[1][2]
tn = cm[0][0]+cm[0][1]+cm[1][0]+cm[1][1]
print("For Virginica \n")
print("Accuracy    :",accuracy_cm(tp,fn,fp,tn))
print("Precision   :",precision_cm(tp,fn,fp,tn))
print("Recall      :",recall_cm(tp,fn,fp,tn))
print("F1-Score    :",f1_score(tp,fn,fp,tn))
print("Error rate  :",error_rate_cm(tp,fn,fp,tn))
```

Practical 7:-
Text Analytics
1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.

```
import nltk
nltk.download('punkt')

#Tokenization
from nltk import word_tokenize, sent_tokenize
sent="Sachin is considered to be one of the greatest cricket player. Virat is the captain of tthe Indian cricket team"
print(word_tokenize(sent))
print(sent_tokenize(sent))

#Stop Wrods Removal
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
stop_words=stopwords.words('english')
print(stop_words)

token=word_tokenize(sent)
cleaned_token=[]
for word in token:
   if word not in stop_words:
      cleaned_token.append(word)
print("This is the unclean version: ",token)
print("This is the cleaned version: ",cleaned_token)

words=[cleaned_token.lower() for cleaned_token in cleaned_token if cleaned_token.isalpha()]
print(words)

#Steamming
from nltk.stem import PorterStemmer
stemmer=PorterStemmer()
port_stemmer_output=[stemmer.stem(words) for words in words]
print(port_stemmer_output)

#Lemmatization
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
lemmatizer=WordNetLemmatizer()
lemmatizer_output = [lemmatizer.lemmatize(word) for word in words]
print(lemmatizer_output)

#POS Tagging
from nltk import pos_tag
import nltk
nltk.download('averaged_perceptron_tagger')
token=word_tokenize(sent)
cleaned_token=[]
for word in token:
   if word not in stop_words:
      cleaned_token.append(word)
tagged=pos_tag(cleaned_token)
print(tagged)
```

2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd

vectorizer = TfidfVectorizer(analyzer="word", norm=None, use_idf=True, smooth_idf=True)
Mat=vectorizer.fit(docs)
print(Mat.vocabulary_)

vectorizer = TfidfVectorizer()
tfidfMat = vectorizer.fit_transform(docs)

print(tfidfMat)

features_name=vectorizer.get_feature_names_out()
print(features_name)

dense=tfidfMat.todense()
denselist=dense.tolist()
df=pd.DataFrame(denselist,columns=features_name)
df

features_name = sorted(vectorizer.get_feature_names_out())
docList=['Doc 1','Doc 2','Doc 3','Doc 4']
skDocsIfIdfdf=pd.DataFrame(tfidfMat.todense(),index=sorted(docList), columns=features_name)
print(skDocsIfIdfdf)

csim=cosine_similarity(tfidfMat,tfidfMat)
csimDf = pd.DataFrame(csim, index=sorted(docList), columns=sorted(docList))
print(csimDf)
```

Practical 8:-

Data Visualization I

1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn libraryto see if we can find any patterns in the data.

2. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.

```
import seaborn as sns
import pandas as pd
titanic=sns.load_dataset("titanic")
titanic

titanic.info()

x=titanic["fare"]
x

titanic.describe()

titanic.info()

titanic_cleaned = titanic.drop(['pclass', 'embarked', 'deck', 'embark_town'], axis=1)
titanic_cleaned.head(15)

titanic_cleaned.info()

titanic_cleaned.isnull().sum()

sns.histplot(data=titanic,x="fare",bins=8)

sns.histplot(data=titanic,x="fare",bins=20,binwidth=10)

sns.histplot(data=titanic,x="fare",binwidth=10)

sns.histplot(data=titanic,x="fare",bins=20,binwidth=1)

sns.histplot(data=titanic,x="fare",bins=20,binwidth=50)
```

Practical 9:-

Data Visualization II

1. Use the inbuilt dataset 'titanic' as used in the above problem. Plot a box plot for distribution of age with respect to each gender along with the information about whether they survived or not. (Column names : 'sex' and 'age')

2. Write observations on the inference from the above statistics.

```
import seaborn as sns
titanic=sns.load_dataset("titanic")
titanic

titanic.head(10)

titanic.info()

titanic.describe()

titanic.loc[:,["survived","alive"]]

sns.boxplot(x="sex",y="age",data=titanic)

sns.boxplot(x="sex",y="age",data=titanic,hue="survived")
```

Practical 10:-

Data Visualization III

Download the Iris flower dataset or any other dataset into a DataFrame. (e.g.,https://archive.ics.uci.edu/ml/datasets/Iris). Scan the dataset and give the inference as:

1. List down the features and their types (e.g., numeric, nominal) available in the dataset.

```
import numpy as np
import pandas as pd

df = pd.read_csv("Iris1.csv",header=None)
df.head()

column = len(list(df))
column

df.info()

np.unique(df[5])

df.describe()
```

2. Create a histogram for each feature in the dataset to illustrate the feature distributions.

```
fig, axes = plt.subplots(2, 2, figsize=(16, 8))
axes[0,0].set_title("Distribution of First Column")
axes[0,0].hist(df[1]);
axes[0,1].set_title("Distribution of Second Column")
axes[0,1].hist(df[2]);
axes[1,0].set_title("Distribution of Third Column")
axes[1,0].hist(df[3]);
axes[1,1].set_title("Distribution of Fourth Column")
axes[1,1].hist(df[4]);
```

3. Create a box plot for each feature in the dataset. & 4. Compare distributions and identify outliers.

```
x=df[[1, 2, 3, 4]]
x.boxplot()
plt.show();
```

Practical 11:-
Write a code in JAVA for a simple Word Count application that counts the number of occurrences of each word in a given input set using the Hadoop Map-Reduce framework onlocal-standalone set-up.

Practical 12:-
Design a distributed application using Map-Reduce which processes a log file of a system.

Practical 13:-
Write a Simple program in SCALA using Apache Spark framework.

Steps

1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo apt install default-jdk
4. wget https://dlcdn.apache .org/spark/spark-3.5.1/spark-3.5.1-bin-hadoop3.tgz
5. tar xvf spark-3.5.1-bin-hadoop3.tgz
6. sudo my spark-3.5.1-bin-hadoop3/opt/spark
7. nano ~/.bashrc
8. Add the following lines at the end of the files
   Export SPARK_HOME=/opt/spark
   Export PATH=$PATH:
   $SPARK_HOME/bin
   Save and exit
9. To open spark shell enter spark-shell

Code

```
val text = sc.textFile(“my_text.txt”)

val counts = text.flatMap(line=> line.split(“ “)).map(word=>(word,1)).reduceByKey(_+_)

counts.collect
```